Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S3	541	((storage memory)) same (creat$4 mak$4 build$4 generat$4 construct$4) same (link list) same (identif$4 detect$4 determin$4 check$4) same (free empty) same (memory location) and (pointer) and (@ad<"20011221" @prad<"20011221" @rlad<"20011221")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/08/10 10:48
S4	388	((storage memory)) same (creat$4 mak$4 build$4 generat$4 construct$4) same (link list) same (identif$4 detect$4 determin$4 check$4) same (free empty) same (memory location) and((splic$4 merg$4 join$4 unit$4 link$4) with (memory location)) and (pointer) and (@ad<"20011221" @prad<"20011221" @rlad<"20011221")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/08/10 10:50
S5	85	((storage memory)) same (creat$4 mak$4 build$4 generat$4 construct$4) with (link list) same (identif$4 detect$4 determin$4 check$4) with (free empty) same (memory location) and((splic$4 merg$4 join$4 unit$4 link$4) with (memory location)) and (pointer) and (@ad<"20011221" @prad<"20011221" @rlad<"20011221")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/08/10 12:17
S7	2	koob.in. and splicing	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/08/10 11:20
S8	20	((storage memory)) same (creat$4 mak$4 build$4 generat$4 construct$4) with (link list) same (identif$4 detect$4 determin$4 check$4) with (free empty) same (memory location) and((splic$4 merg$4 join$4 unit$4 link$4) with (memory location)) and (pointer) and (tail same head) and (@ad<"20011221" @prad<"20011221" @rlad<"20011221")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/08/10 12:22

Terms used **free link lists**

Found **13,941** of **158,639**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Shared momory objects: Lock-free linked lists and skip lists

Mikhail Fomitchev, Eric Ruppert

July 2004 **Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing**

Full text available: pdf(225.72 KB) Additional Information: full citation, abstract, references, index terms

Lock-free shared data structures implement distributed objects without the use of mutual exclusion, thus providing robustness and reliability. We present a new lock-free implementation of singly-linked lists. We prove that the worst-case amortized cost of the operations on our linked lists is linear in the length of the list plus the contention, which is better than in previous lock-free implementations of this data structure. Our implementation uses backlinks that are set when a node is deleted ...

Keywords: amortized analysis, analysis, distributed, efficient, fault-tolerant, linked list, lock-free, skip list

2 Lock-free linked lists using compare-and-swap

John D. Valois

August 1995 **Proceedings of the fourteenth annual ACM symposium on Principles of distributed computing**

Full text available: pdf(902.29 KB) Additional Information: full citation, references, citings, index terms

3 Shadowed management of free disk pages with a linked list

Matthew S. Hecht, John D. Gabbe

December 1983 **ACM Transactions on Database Systems (TODS)**, Volume 8 Issue 4

Full text available: pdf(877.39 KB) Additional Information: full citation, abstract, references, index terms

We describe and prove correct a programming technique using a linked list of pages for managing the free disk pages of a file system where shadowing is the recovery technique. Our technique requires a window of only two pages of main memory for accessing and maintaining the free list, and avoids wholesale copying of free-list pages during a checkpoint or recover operation.

Keywords: checkpoint, dynamic storage allocation, file system, recovery, shadowing, storage management

4 Split-ordered lists: lock-free extensible hash tables

Ori Shalev, Nir Shavit

July 2003 **Proceedings of the twenty-second annual symposium on Principles of distributed computing**

Full text available: pdf(1.06 MB) Additional Information: full citation, abstract, references, citings, index terms

We present the first lock-free implementation of an extensible hash table running on current architectures. It provides concurrent insert, delete, and search operations with an expected $O(1)$ cost. It consists of very simple code, easily implementable using only load, store, and compare-and-swap operations. The new mathematical structure at the core of our algorithm is *recursive split-ordering*, a way of ordering elements in a linked list so that they can be repeatedly "split" using ...

Keywords: Compare-and-Swap, Concurrent Data Structures, Hash Table, Non-blocking Synchronization, Real-Time

5 Session 3: High performance dynamic lock-free hash tables and list-based sets

Maged M. Michael

August 2002 **Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures**

Full text available: pdf(238.11 KB) Additional Information: full citation, abstract, references, citings, index terms

Lock-free (non-blocking) shared data structures promise more robust performance and reliability than conventional lock-based implementations. However, all prior lock-free algorithms for sets and hash tables suffer from serious drawbacks that prevent or limit their use in practice. These drawbacks include size inflexibility, dependence on atomic primitives not supported on any current processor architecture, and dependence on highly-inefficient or blocking memory management techniques.Building on ...

6 Garbage Collection of Linked Data Structures

Jacques Cohen

September 1981 **ACM Computing Surveys (CSUR)**, Volume 13 Issue 3

Full text available: pdf(2.32 MB) Additional Information: full citation, references, citings, index terms

7 Implementing wait-free objects on priority-based systems

James H. Anderson, Srikanth Ramamurthy, Rohit Jain

August 1997 **Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing**

Full text available: pdf(1.19 MB) Additional Information: full citation, references, citings, index terms

8 Session 1: Safe memory reclamation for dynamic lock-free objects using atomic reads and writes

Maged M. Michael

July 2002 **Proceedings of the twenty-first annual symposium on Principles of distributed computing**

Full text available: pdf(1.13 MB) Additional Information: full citation, abstract, references, citings

A major obstacle to the wide use of lock-free data structures, despite their many performance and reliability advantages, is the absence of a practical lock-free method for reclaiming the memory of dynamic nodes removed from dynamic lock-free objects for arbitrary reuse.The only prior lock-free memory reclamation method depends on the DCAS atomic primitive, which is not supported on any current processor architecture. Other memory management methods are blocking, require special operating system ...

9 Scalable lock-free dynamic memory allocation

Maged M. Michael

June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation**, Volume 39 Issue 6

Full text available: pdf(213.94 KB) Additional Information: full citation, abstract, references, citings, index terms

Dynamic memory allocators (malloc/free) rely on mutual exclusion locks for protecting the consistency of their shared data structures under multithreading. The use of locking has many disadvantages with respect to performance, availability, robustness, and programming flexibility. A lock-free memory allocator guarantees progress regardless of whether some threads are delayed or even killed and regardless of scheduling policies. This paper presents a completely lock-free memory allocator. It uses ...

Keywords: async-signal-safe, availability, lock-free, malloc

10 A method for implementing lock-free shared-data structures

Greg Barnes

August 1993 **Proceedings of the fifth annual ACM symposium on Parallel algorithms and architectures**

Full text available: pdf(978.61 KB) Additional Information: full citation, references, citings, index terms

11 Compact Encodings of List Structure

Daniel G. Bobrow, Douglas W. Clark

October 1979 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 1 Issue 2

Full text available: pdf(1.40 MB) Additional Information: full citation, abstract, references, citings, index terms

List structures provide a general mechanism for representing easily changed structured data, but can introduce inefficiencies in the use of space when fields of uniform size are used to contain pointers to data and to link the structure. Empirically determined regularity can be exploited to provide more space-efficient encodings without losing the flexibility inherent in list structures. The basic scheme is to provide compact pointer fields big enough to accommodate most values that occur i ...

12 Architectural mechanisms to support sparse vector processing

R. N. Ibbett, T. M. Hopkins, K. I. M. McKinnon

April 1989 **ACM SIGARCH Computer Architecture News , Proceedings of the 16th annual international symposium on Computer architecture**, Volume 17 Issue 3

Full text available: pdf(1.11 MB) Additional Information: full citation, abstract, references, citings, index terms

We discuss the algorithmic steps involved in common sparse matrix problems, with particular emphasis on linear programming by the revised simplex method. We then propose new architectural mechanisms which are being built into an experimental machine, the Edinburgh Sparse Processor, and which enable vector instructions to operate efficiently on sparse vectors stored in compressed form. Finally, we review the use of these new mechanisms on the linear programming problem.

13 Formalization of operations and function definitions in a functional programming language for data structures

John C. Thompson, Reza Sanati-Mehrizy

February 1987 **Proceedings of the 15th annual conference on Computer Science**

Full text available: pdf(774.63 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a method for formalization of operations on certain class of syntactically represented linked Data Structures and then defines a set of functions to perform the operations. The formalization will be done by designing a set of transformation rules which are based on the grammars generating these Data Structures. Rooted in formal language theory, this method of formalization provides a basis for mathematical

demonstration of the correctness of the operations. The function ...

14 On the time required for retention

D. M. Berry, L. Chirica, J. B. Johnston, D. F. Martin, A. Sorkin

November 1973 **Proceedings of the ACM-IEEE symposium on High-level-language computer architecture**

Full text available: pdf(1.21 MB) Additional Information: full citation, abstract, references, citings, index terms

In block structured languages, upon entry to a block or procedure, storage is allocated for the identifiers declared in the block or procedure. There are two choices as to when to deallocate this storage: 1) upon exit from the block or procedure, 2) when the storage becomes inaccessible. In the deletion strategy, storage for blocks and procedures is allocated and deallocated in a last-in-first-out order. Therefore, storage for blocks and procedures can be ma ...

15 On the time required for retention

D. M. Berry, L. Chirica, J. B. Johnston, D. F. Martin, A. Sorkin

November 1973 **ACM SIGPLAN Notices , Proceedings of a symposium on High-level-language computer architecture**, Volume 8 Issue 11

Full text available: pdf(1.19 MB) Additional Information: full citation, references, citings

16 Simple, fast, and practical non-blocking and blocking concurrent queue algorithms

Maged M. Michael, Michael L. Scott

May 1996 **Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing**

Full text available: pdf(860.15 KB) Additional Information: full citation, references, citings, index terms

Keywords: compare_and_swap, concurrent queue, lock-free, multiprogramming, non-blocking

17 Design and evaluation of a DRAM-based shared memory ATM switch

Tzi-cker Chiueh, Srinidhi Varadarajan

June 1997 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 1997 ACM SIGMETRICS international conference on Measurement and modeling of computer systems**, Volume 25 Issue 1

Full text available: pdf(1.55 MB) Additional Information: full citation, abstract, references, citings, index terms

Beluga is a single-chip switch architecture specifically targeted at local area ATM networks, and it features three architectural innovations. First, an interconnection hierarchy composed of multiple switching fabrics is built into the chip to provide both low-latency cell transfer when the traffic is light and low cell drop rate under heavy load. Secondly, to improve silicon efficiency, Beluga is based on shared memory architecture, and the buffers are implemented using DRAM rather than ...

18 A linear-time scheme for version reconstruction

Lin Yu, Daniel J. Rosenkrantz

May 1994 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 16 Issue 3

Full text available: pdf(1.47 MB) Additional Information: full citation, abstract, references, citings, index terms, review

An efficient scheme to store and reconstruct versions of sequential files is presented. The reconstruction scheme involves building a data structure representing a complete version, and then successively modifying this data structure by applying a sequence of specially

formatted differential files to it. Each application of a differential file produces a representation of an intermediate version, with the final data structure representing the requested version. The scheme uses a ...

Keywords: data structures, database systems, differential files, document preparation, software systems, textual objects, version control

19 Garbage collecting the Internet: a survey of distributed garbage collection

Saleh E. Abdullahi, Graem A. Ringwood

September 1998 **ACM Computing Surveys (CSUR)**, Volume 30 Issue 3

Full text available: pdf(337.65 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Internet programming languages such as Java present new challenges to garbage-collection design. The spectrum of garbage-collection schema for linked structures distributed over a network are reviewed here. Distributed garbage collectors are classified first because they evolved from single-address-space collectors. This taxonomy is used as a framework to explore distribution issues: locality of action, communication overhead and indeterministic communication latency.

Keywords: automatic storage reclamation, distributed, distributed file systems, distributed memories, distributed object-oriented management, memory management, network communication, object-oriented databases, reference counting

20 Efficient implementation of the first-fit strategy for dynamic storage allocation

R. P. Brent

July 1989 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 11 Issue 3

Full text available: pdf(1.05 MB) Additional Information: full citation, abstract, references, citings, index terms, review

We describe an algorithm that efficiently implements the first-fit strategy for dynamic storage allocation. The algorithm imposes a storage overhead of only one word per allocated block (plus a few percent of the total space used for dynamic storage), and the time required to allocate or free a block is O(log W), where W is the maximum number of words allocated dynamically. The algorithm is faster than many commonly used algorithms, especia ...